



Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie  
Wydział Informatyki, Elektroniki i Telekomunikacji  
Instytut Informatyki

## Języki formalne i kompilatory

Projekt zaliczeniowy

*Translator języka  $\LaTeX$  do HTML*

**Bartosz Kordek Grzegorz Zacharski**

Informatyka

Studia niestacjonarne

Kraków, 2021

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Pliki tworzące projekt . . . . .	2
1.2	Uruchamianie . . . . .	2
<b>2</b>	<b>Obsługiwany zakres translatora</b>	<b>4</b>
2.1	Tekst . . . . .	4
2.2	Formatowanie tekstu . . . . .	4
2.3	Tabela . . . . .	4
2.4	Wyliczenie . . . . .	5
2.5	Grafika . . . . .	5
2.6	Hiperłącze . . . . .	5
2.7	Sekcja, podsekcja, podpodsekcja . . . . .	5
<b>3</b>	<b>Gramatyka</b>	<b>6</b>
3.1	Lexer . . . . .	6
3.2	Parser . . . . .	8
3.2.1	Obsługa tekstu . . . . .	9
3.2.2	Formatowanie tekstu . . . . .	9
3.2.3	Tabela . . . . .	11
3.2.4	Wyliczenie . . . . .	12
3.2.5	Grafika . . . . .	13
3.2.6	Hiperłącze . . . . .	14
<b>4</b>	<b>Uzasadnienie wyboru generatora</b>	<b>15</b>
<b>5</b>	<b>Napotkane problemy</b>	<b>16</b>
5.1	Przejsie do nowej linii . . . . .	16
5.2	Translacja tabeli . . . . .	16

# Rozdział 1

## Wstęp

Celem projektu było zaimplementowanie translatora języka  $\text{\LaTeX}$  do HTML. W tym celu skorzystaliśmy z gotowej biblioteki `PLY` [3] do skanowania i parsowania napisanej w języku Python. Kod odpowiadający za wykonanie translacji podzieliliśmy na dwie części: lekser (skanowanie) i parser (parsowanie).

Do wykonania projektu skorzystaliśmy również z oficjalnej dokumentacji języka  $\text{\LaTeX}$  [2] oraz języka HTML [1].

### 1.1 Pliki tworzące projekt

Translator został zaimplementowany w języku Python 3. Projekt składa się z następujących plików:

- `main.py` - plik wejściowy programu, odpowiada za uruchomienie interpretera, wyświetlanie pomocniczych informacji w trybie debugowania oraz za uruchomienie odpowiedniego trybu działania programu
- `theLexer.py` - lexer, który ma za zadanie zwrócić tokeny po przeanalizowaniu kodu dokumentu latex
- `theParser.py` - parser, który ma za zadanie stworzyć abstrakcyjne drzewo syntaktyczne oraz zwrócić kod html
- `example.tex` - przykładowy plik testowy, który zostanie przetłumaczony do języka html

### 1.2 Uruchamianie

W celu uruchomienia translatora trzeba mieć zainstalowany interpreter języka Python 3 oraz w folderze projektu wpisać w terminalu komendę pokazaną na listingu 1.1:

```
1 python3 main.py -i example -o example
```

Listing 1.1: Uruchomienie

Parametr *i* przyjmuje nazwę pliku do translacji, natomiast paramentr *o* przyjmuje nazwę pliki html, który zostanie stworzony podczas działania programu.

# Rozdział 2

## Obsługiwany zakres translatora

### 2.1 Tekst

Nie wyszczególniono specjalnych wymagań, które musi spełniać zwykły tekst w języku  $\text{\LaTeX}$ , aby został przetłumaczony na język HTML.

### 2.2 Formatowanie tekstu

Translator obsługuje następujące znaczniki formatu  $\text{\LaTeX}$  odpowiadające na formatowanie tekstu:

- $\text{\texttt{\textbackslash textbf}}$  - pogrubienie
- $\text{\texttt{\textbackslash textit}}$  - kursywa
- $\text{\texttt{\textbackslash underline}}$  - podkreślenie
- $\text{\texttt{\textbackslash centerline}}$  - wyśrodkowanie
- $\text{\texttt{\textbackslash newline}}$  - nowa linia
- $\text{\texttt{\textbackslash title}}$  - tytuł

### 2.3 Tabela

W celu przetłumaczenia tabeli z języka  $\text{\LaTeX}$  na język HTML tabela powinna spełniać poniższe założenia:

- musi być zadeklarowana poleceniem  $\text{\texttt{\textbackslash begin\{tabular\}}}$  oraz zakończona  $\text{\texttt{\textbackslash end\{tabular\}}}$
- po deklaracji  $\text{\texttt{\textbackslash begin\{tabular\}}}$  musi zostać od razu zadeklarowane obramowanie wg standardu  $\text{\LaTeX}$ , np.  $\text{\texttt{\textbackslash begin\{tabular\}\{c c c\}}}$

- tabele z obramowaniem muszą zostać zadeklarowane wg `\begin{tabular}{| c | c | c |}` - ilość kolumn dowolna
- tabele bez obramowania muszą zostać zadeklarowane wg `\begin{tabular}{c c c}` - ilość kolumn dowolna

## 2.4 Wyliczenie

Program obsługuje wyliczenie uporządkowane i nieuporządkowane:

- `\begin{enumerate}` - uporządkowane
- `\begin{itemize}` - nieuporządkowane

## 2.5 Grafika

Grafikę w tekście możemy zadeklarować wyłącznie używając polecenia `\includegraphics` podając ścieżkę do zdjęcia np: `\includegraphics{corgi.jpg}`.

## 2.6 Hiperłącze

Hiperłącze w deklarujemy wyłącznie poprzez użycie `\url` np. `\url{https://github.com/bartoszkordek/AGH-Języki-formalne-i-kompilatory}`

## 2.7 Sekcja, podsekcja, podpodsekcja

Sekcje, podsekcje oraz podpodsekcje deklarujemy w ten sam sposób jak w języku  $\text{\LaTeX}$  np. `\subsection{Przykładowa podsekcja}`.

# Rozdział 3

## Gramatyka

### 3.1 Lexer

Na listingu 3.1 zostały przedstawione tokeny obsługiwane przez nasz translator.

```
1  tokens = (  
2      'AUTHOR',  
3      'BEGIN_DOCUMENT',  
4      'BEGIN_OLIST',  
5      'BEGIN_ULIST',  
6      'BEGIN_TABULAR',  
7      'BOLD',  
8      'CENTERLINE',  
9      'CHAPTER',  
10     'COLUMN_DIVIDER',  
11     'COLUMN_PATTERN_BORDERLESS',  
12     'COLUMN_PATTERN_BORDERED',  
13     'DOCUMENTCLASS',  
14     'END_DOCUMENT',  
15     'END_OLIST',  
16     'END_ULIST',  
17     'END_TABULAR',  
18     'GRAPHICS_PATH',  
19     'HLINE',  
20     'INCLUDE_GRAPHICS',  
21     'ITALIC',  
22     'ITEM',  
23     'LBRACE',  
24     'NEW_LINE',  
25     'NULL',  
26     'PARAGRAPH',  
27     'RBRACE',  
28     'ROW_END',  
29     'SECTION',
```

```

30     'SUBSECTION',
31     'SUBSUBSECTION',
32     'TEXT',
33     'TITLE',
34     'UNDERLINE',
35     'URL',
36     'USE_PACKAGE',
37 )

```

Listing 3.1: Tokeny

Tokeny na listingu 3.1 odpowiadają następującym wyrażeniom regularnym przedstawionym na listingu 3.2:

```

1  t_AUTHOR = r'\\author'
2  t_BEGIN_DOCUMENT = r'\\begin\\{document\\}'
3  t_BEGIN_OLIST = r'\\begin\\{enumerate\\}'
4  t_BEGIN_ULIST = r'\\begin\\{itemize\\}'
5  t_BEGIN_TABULAR = r'\\begin\\{tabular\\}'
6  t_BOLD = r'\\textbf'
7  t_CENTERLINE = r'\\centerline'
8  t_CHAPTER = r'\\chapter'
9  t_COLUMN_DIVIDER = r'&'
10 t_COLUMN_PATTERN_BORDERLESS = r'\\{[lcr](\\s[lcr])*\\}'
11 t_COLUMN_PATTERN_BORDERED = r'\\{(\\|\\s[lcr]\\s)+\\|\\}'
12 t_DOCUMENTCLASS = r'\\documentclass.*'
13 t_END_DOCUMENT = r'\\end\\{document\\}'
14 t_END_OLIST = r'\\end\\{enumerate\\}'
15 t_END_ULIST = r'\\end\\{itemize\\}'
16 t_END_TABULAR = r'\\end\\{tabular\\}'
17 t_GRAPHICS_PATH = r'\\graphicspath'
18 t_INCLUDE_GRAPHICS = r'\\includegraphics'
19 t_ITALIC = r'\\textit'
20 t_ITEM = r'\\item'
21 t_LBRACE = r'\\{'
22 t_NEW_LINE = r'\\newline'
23 t_NULL = r'\\0'
24 t_PARAGRAPH = r'\\paragraph'
25 t_RBRACE = r'\\}'
26 t_ROW_END = r'\\\\'
27 t_SECTION = r'\\section'
28 t_SUBSECTION = r'\\subsection'
29 t_SUBSUBSECTION = r'\\subsubsection'
30 t_TEXT = r'[\\w\\d\\.,!?@#\\/\\'\\<>\\(\\)\\-+=\\/\\^\\*\\:;|\\[\\]]+'
31 t_TITLE = r'\\title'
32 t_UNDERLINE = r'\\underline'
33 t_URL = r'\\url'

```



```
34 t_USE_PACKAGE = r'\\usepackage.*'
```

Listing 3.2: Wyrażenia regularne

Definicje tokenów nowej linii, komentarza, błędów oraz tokenów ignorowanych zostały przedstawione na listingu 3.3.

```
1 def t_newline(self, t):
2     r'\n+'
3     t.lexer.lineno += t.value.count("\n")
4
5 def t_comment(self, t):
6     r'%.*\n'
7     pass
8
9 def t_hline(self, t):
10    r'\\hline'
11    pass
12
13 t_ignore = ' '
14
15 def t_error(self, t):
16    print("Illegal character '%s'" % t.value[0])
17    t.lexer.skip(1)
```

Listing 3.3: Pozostałe tokeny

Należy zauważyć, że cała zawartość między znacznikiem "%" w formacie L<sup>A</sup>T<sub>E</sub>X, odpowiadającemu początkowi komentarza, do nowej linii, zostaje pominięta.

## 3.2 Parser

Każdy z dokumentów HTML'a musi zawierać znaczniki typowe dla tego formatu, których nie znajdziemy w dokumentach L<sup>A</sup>T<sub>E</sub>X. Są to: *html*, *head* oraz *body*. Jako odpowiednik sekcji *head* przyjęliśmy sekcję preambuly w której znajdują się między innymi takie polecenia jak *usepackage*. Natomiast jak odpowiednik sekcji *body* odpowiada fragment kodu L<sup>A</sup>T<sub>E</sub>X objęty poleceniami `\begin{document}` oraz `\end{document}`.

```
1 def p_statement_preamble(self, p):
2     'statement : DOCUMENTCLASS head body'
3     p[0] = '<!DOCTYPE html>' + '\n<html lang="en">' + p[2]
4         + p[3] + '\n</html>'
5
6 def p_header(self, p):
7     '''head : USE_PACKAGE head
8         | USE_PACKAGE'''
9     p[0] = '\n<head>' + '<meta charset="UTF-8"/>' + '\n</head>'
10
```

```

11     def p_body(self, p):
12         'body : BEGIN_DOCUMENT expression END_DOCUMENT'
13         p[0] = '\n<body>\n' + p[2] + '\n</body>'

```

Listing 3.4: Preambuła

### 3.2.1 Obsługa tekstu

```

1     def p_expression_text(self, p):
2         '''expression : TEXT expression
3             | TEXT'''
4         if len(p) == 3:
5             p[0] = p[1] + " " + p[2]
6         else:
7             p[0] = p[1]

```

Listing 3.5: Tekst

### 3.2.2 Formatowanie tekstu

Nasz translator umożliwia pogrubienie, kursywę, podkreślenie, wyśrodkowanie tekstu oraz utworzenie paragrafu. Możliwe jest mieszanie stylów formatowania, np. pogrubienie z podkreśleniem. Według nowych zaleceń, do pogrubienia tekstu w HTML'u powinno się stosować znacznik *<strong>*, a do kursywy *<em>*.

```

1     def p_expression_bold(self, p):
2         '''expression : BOLD LBRACE expression RBACE expression
3             | BOLD LBRACE expression RBACE'''
4         if len(p) == 6:
5             p[0] = '<strong>' + p[3] + '</strong>' + p[5]
6         else:
7             p[0] = '<strong>' + p[3] + '</strong>'
8
9     def p_expression_italic(self, p):
10        '''expression : ITALIC LBRACE expression RBACE expression
11            | ITALIC LBRACE expression RBACE'''
12        if len(p) == 6:
13            p[0] = '<em>' + p[3] + '</em>' + p[5]
14        else:
15            p[0] = '<em>' + p[3] + '</em>'
16
17    def p_expression_underline(self, p):
18        '''expression : UNDERLINE LBRACE expression RBACE expression
19            | UNDERLINE LBRACE expression RBACE'''
20        if len(p) == 6:
21            p[0] = '<u>' + p[3] + '</u>' + p[5]

```

```

22         else:
23             p[0] = '<u>' + p[3] + '</u>'
24
25     def p_expression_centerline(self, p):
26         '''expression : CENTERLINE LBRACE expression RBRACE expression
27             | CENTERLINE LBRACE expression RBRACE'''
28         if len(p) == 6:
29             p[0] = '<center>' + p[3] + '</center>' + p[5]
30         else:
31             p[0] = '<center>' + p[3] + '</center>'
32
33     def p_expression_paragraph(self, p):
34         '''expression : PARAGRAPH LBRACE expression RBRACE expression
35             | PARAGRAPH LBRACE expression RBRACE'''
36         if len(p) == 6:
37             p[0] = '<p>' + p[3] + '</p>' + p[5]
38         else:
39             p[0] = '<p>' + p[3] + '</p>'

```

Listing 3.6: Formatowanie - pogrubienie, kursywa, podkreślenie, wyśrodkowanie tekstu

Kolejną funkcjonalnością jest możliwość obsługi rozdziałów, sekcji i podsekcji parsując znaczniki *chapter*, *section*, *subsection* oraz *subsubsection*.

```

1     def p_expression_chapter(self, p):
2         '''expression : CHAPTER LBRACE expression RBRACE expression
3             | CHAPTER LBRACE expression RBRACE'''
4         if len(p) == 6:
5             p[0] = '<h1>' + p[3] + '</h1>' + p[5]
6         else:
7             p[0] = '<h1>' + p[3] + '</h1>'
8
9     def p_expression_section(self, p):
10        '''expression : SECTION LBRACE expression RBRACE expression
11            | SECTION LBRACE expression RBRACE'''
12        if len(p) == 6:
13            p[0] = '<h2>' + p[3] + '</h2>' + p[5]
14        else:
15            p[0] = '<h2>' + p[3] + '</h2>'
16
17    def p_expression_subsection(self, p):
18        '''expression : SUBSECTION LBRACE expression RBRACE expression
19            | SUBSECTION LBRACE expression RBRACE'''
20        if len(p) == 6:
21            p[0] = '<h3>' + p[3] + '</h3>' + p[5]
22        else:
23            p[0] = '<h3>' + p[3] + '</h3>'
24

```

```

25     def p_expression_subsubsection(self, p):
26         '''expression : SUBSUBSECTION LBRACE expression RBRACE expression
27                        | SUBSUBSECTION LBRACE expression RBRACE'''
28         if len(p) == 6:
29             p[0] = '<h4>' + p[3] + '</h4>' + p[5]
30         else:
31             p[0] = '<h4>' + p[3] + '</h4>'

```

Listing 3.7: Rozdziały, sekcje i podsekcje

Przejsście do nowej linii (hard break) jest obsługiwane poleceniem *newline*.

```

1     def p_expression_newline(self, p):
2         '''expression : NEW_LINE expression
3                        | NEW_LINE'''
4         if len(p) == 3:
5             p[0] = '<br/>' + p[2]
6         else:
7             p[0] = p[1]

```

Listing 3.8: Nowa linia

Translator parsuje również znacznik *title* odpowiadający za utworzenie tytułu.

```

1     def p_title(self, p):
2         'expression : TITLE LBRACE TEXT RBRACE'
3         p[0] = '<i>' + p[3] + '</i>'

```

Listing 3.9: Tytuł

### 3.2.3 Tabela

#### Z obramowaniem

```

1     def p_expression_table_bordered(self, p):
2         '''expression : BEGIN_TABULAR COLUMN_PATTERN_BORDERED
3            tablerowbordered END_TABULAR expression | BEGIN_TABULAR
4            COLUMN_PATTERN_BORDERED tablerowbordered END_TABULAR'''
5         if len(p) == 6:
6             p[0] = '<table style="border: 1px solid black; border-collapse:
7 collapse;">' + p[3] + '</table>' + p[5]
8         else:
9             p[0] = '<table style="border: 1px solid black; border-collapse:
10 collapse;">' + p[3] + '</table>'
11
12     def p_tablerowbordered(self, p):
13         '''tablerowbordered : tablecolumnbordered ROW_END tablerowbordered |
14            tablecolumnbordered'''
15         if len(p) == 4:
16             p[0] = '<tr>' + p[1] + '</tr>' + p[3]

```

```

12         else:
13             p[0] = '<tr>' + p[1] + '</tr>'
14
15     def p_tablecolumnbordered(self, p):
16         '''tablecolumnbordered : expression COLUMN_DIVIDER
17         tablecolumnbordered | expression'''
18         if len(p) == 4:
19             p[0] = '<td style="border: 1px solid black; border-collapse:
20 collapse;">' + p[1] + '</td>' + p[3]
21         else:
22             p[0] = '<td style="border: 1px solid black; border-collapse:
23 collapse;">' + p[1] + '</td>'

```

Listing 3.10: Tabela z obramowaniem

## Bez obramowania

```

1     def p_expression_table_borderless(self, p):
2         '''expression : BEGIN_TABULAR COLUMN_PATTERN_BORDERLESS
3         tablerowborderless END_TABULAR expression | BEGIN_TABULAR
4         COLUMN_PATTERN_BORDERLESS tablerowborderless END_TABULAR'''
5         if len(p) == 6:
6             p[0] = '<table>' + p[3] + '</table>' + p[5]
7         else:
8             p[0] = '<table>' + p[3] + '</table>'
9
10    def p_tablerowborderless(self, p):
11        '''tablerowborderless : tablecolumnborderless ROW_END
12        tablerowborderless | tablecolumnborderless'''
13        if len(p) == 4:
14            p[0] = '<tr>' + p[1] + '</tr>' + p[3]
15        else:
16            p[0] = '<tr>' + p[1] + '</tr>'
17
18    def p_tablecolumnborderless(self, p):
19        '''tablecolumnborderless : expression COLUMN_DIVIDER
20        tablecolumnborderless | expression'''
21        if len(p) == 4:
22            p[0] = '<td>' + p[1] + '</td>' + p[3]
23        else:
24            p[0] = '<td>' + p[1] + '</td>'

```

Listing 3.11: Tabela bez obramowania

### 3.2.4 Wyliczenie

Konstrukcja parsera wyliczeń umożliwia wykonywanie zagnieżdżeń.

## Uporządkowane

```
1 def p_expression_ordered_list(self, p):
2     '''expression : BEGIN_OLIST listitems END_OLIST expression
3       | BEGIN_OLIST listitems END_OLIST'''
4     if len(p) == 5:
5         p[0] = '\n<ol>' + p[2] + '\n</ol>' + p[4]
6     else:
7         p[0] = '\n<ol>' + p[2] + '\n</ol>'
```

Listing 3.12: Wyliczenie uporządkowane

## Nieuporządkowane

```
1 def p_expression_unordered_list(self, p):
2     '''expression : BEGIN_ULIST listitems END_ULIST expression
3       | BEGIN_ULIST listitems END_ULIST'''
4     if len(p) == 5:
5         p[0] = '\n<ul>' + p[2] + '\n</ul>' + p[4]
6     else:
7         p[0] = '\n<ul>' + p[2] + '\n</ul>'
```

Listing 3.13: Wyliczenie nieuporządkowane

### 3.2.5 Grafika

Umieszczenie grafiki jest możliwe dzięki znacznikowi *includegraphics* w  $\text{\LaTeX}$ , który jest parsowany na znacznik `<img>` w HTML, gdzie atrybut *src* stanowi ścieżka do pliku umieszczona w nawiasach wąsatych w dokumencie  $\text{\LaTeX}$ .

```
1 def p_expression_includegraphics(self, p):
2     '''expression : INCLUDE_GRAPHICS TEXT LBRACE expression RBRACE
3       expression
4       | INCLUDE_GRAPHICS LBRACE expression RBRACE expression
5       | INCLUDE_GRAPHICS LBRACE expression RBRACE'''
6     if len(p) == 7:
7         attributes = p[2][1:-1]
8         p[0] = ''''''
9     + p[6]
10    elif len(p) == 6:
11        p[0] = '''''' + p[5]
12    else:
13        p[0] = ''''''
```

Listing 3.14: Grafika

### 3.2.6 Hiperłącze

Zamieszczenie hiperłącza w formacie  $\text{\LaTeX}$  jest możliwe dzięki znacznikowi *url*, zawierającego w nawiasach wąsatych adres do strony. Parsowany jest on na HTML'owy znacznik  $\langle a \rangle$  z atrybutem *href* zawierającego adres.

```
1  def p_expression_url(self, p):
2      '''expression : URL LBRACE expression RBRACE expression
3                      | URL LBRACE expression RBRACE'''
4      if len(p) == 6:
5          p[0] = '<a href=' + p[3] + '>' + p[3] + '</a>' + p[5]
6      else:
7          p[0] = '<a href=' + p[3] + '>' + p[3] + '</a>'
```

Listing 3.15: Hiperłącze

## Rozdział 4

# Uzasadnienie wyboru generatora

Podstawowym wymaganiem w projekcie była konieczność zastowania generatora parserów i skanerów. Z zaproponowanych na zajęciach bibliotek (PLY, SLY oraz ANTLR) zdecydowaliśmy się na wybór biblioteki PLY [3], ze względu na jej popularność stosowania, dostępność i wysoką jakość dokumentacji. Ponadto chcieliśmy również zdobyć kolejne doświadczenie w implementacji kodu z wykorzystaniem języka Python. Na niekorzyść wykorzystania biblioteki SLY przemawiała niska jakość dokumentacji oraz brak przykładów.



# Rozdział 5

## Napotkane problemy

Podczas realizacji projektu napotkaliśmy następujące problemy do rozwiązania:

- przejście do nowej linii
- translacja tabeli

### 5.1 Przejście do nowej linii

W języku  $\text{\LaTeX}$  przejście do nowej linii jest możliwe dzięki następujący sposób: wykorzystując znacznik `\newline`, podwójnemu backslashowi, oraz wykonaniem podwójnego prześcia do nowej linii za pomocą klawisza Enter. W rezultacie chcieliśmy otrzymać znacznik `<br>` w HTMLu. Nie napotkaliśmy większych problemów z translacją znacznika `\newline`. Podwójny backslash również działał poprawnie dopóki nie zaimplementowaliśmy translacji tabeli, w której ten sam znacznik oznacza jej koniec. Ze względu na ten problem musieliśmy z niego zrezygnować. Problem nastąpił również z przejściem do nowej linii przy pomocy dwukrotnego wciśnięcia klawisza Enter. W tym przypadku każde przejście do nowej linii było traktowane jako przejście pojedyncze, a podwójne było pomijane. W rezultacie oznaczało to, że nie mogliśmy otrzymać znacznika `<br>` w HTMLu. Po usunięciu definicji tokena `\newline`, znacznik `<br>` był produkowany, lecz program generował ostrzeżenie (brak znajomości tokena) w przypadku wystąpienia pojedynczego przejścia. Plik w formacie HTML był jednak poprawnie generowany. Ze względu na ostrzeżenie, pomysł został porzucony.

### 5.2 Translacja tabeli

Podstawowym problemem był opisany powyżej znacznik przejścia do nowej linii, który w przypadku tabeli oznacza jej koniec. Ponadto tabela jest na tyle złożoną strukturą, że nie jesteśmy w stanie zapewnić obsługi każdego z jej przypadków. Uprościliśmy więc translację do tabeli w pełni obramowanej lub nieobramowanej.

# Listings

1.1	Uruchomienie . . . . .	3
3.1	Tokeny . . . . .	6
3.2	Wyrażenia regularne . . . . .	7
3.3	Pozostałe tokeny . . . . .	8
3.4	Preambuła . . . . .	8
3.5	Tekst . . . . .	9
3.6	Formatowanie - pogrubienie, kursywa, podkreślenie, wyśrodkowanie tekstu . . .	9
3.7	Rozdziały, sekcje i podsekcje . . . . .	10
3.8	Nowa linia . . . . .	11
3.9	Tytuł . . . . .	11
3.10	Tabela z obramowaniem . . . . .	11
3.11	Tabela bez obramowania . . . . .	12
3.12	Wyliczenie uporządkowane . . . . .	13
3.13	Wyliczenie nieuporządkowane . . . . .	13
3.14	Grafika . . . . .	13
3.15	Hiperłącze . . . . .	14

# Bibliografia

- [1] HTML 5.2. *W3C Recommendation*. 2021. URL: <https://www.w3.org/TR/html52/>.
- [2] Overleaf. *Official Documentation*. 2021. URL: <https://www.overleaf.com/learn>.
- [3] PLY. *Official Documentation*. 2021. URL: <https://ply.readthedocs.io/en/latest/>.